Report Number: WUCS-89-26

1989-06-01

A Learning Algorithm for Acyclic Neural Networks

Takayuki Dan Kimura

Backpropagation as learning rule is often confining to multi-layer Perceptions or layered feedforward networks, in which there are no lateral connections among the units of the same layer nor any connections bypassing intermediate layers. We prove algebraically that these restrictions are not necessary, i.e. backpropagation is applicable to any acyclic neural network. Our proof is based on a new formulation of backpropagation called an Acyclic Neural Network (ANN). In ANN, a net is defined as a partially ordered set of processing units where every unit may receive an input value and/or a correction (teaching) value. Therefore, there is no... **Read complete abstract on page 2.**

A Learning Algorithm for Acyclic Neural Networks

Takayuki Dan Kimura

Complete Abstract:

Backpropagation as learning rule is often confining to multi-layer Perceptions or layered feedforward networks, in which there are no lateral connections among the units of the same layer nor any connections bypassing intermediate layers. We prove algebraically that these restrictions are not necessary, i.e. backpropagation is applicable to any acyclic neural network. Our proof is based on a new formulation of backpropagation called an Acyclic Neural Network (ANN). In ANN, a net is defined as a partially ordered set of processing units where every unit may receive an input value and/or a correction (teaching) value. Therefore, there is no need to differentiate between the input, hidden, and output units.

A LEARNING ALGORITHM FOR ACYCLIC NEURAL NETWORKS

Takayuki Dan Kimura

WUCS-89-26

June 1989
(Revised November 1989)

Department of Computer Science
Washington University
Campus Box 1045
One Brookings Drive
Saint Louis, MO 63130-4899

Abstract

Backpropagation as a learning rule is often confined to multi-layer Perceptrons or layered feedforward networks, in which there are no lateral connections among the units of the same layer nor any connections bypassing intermediate layers. We prove algebraically that these restrictions are not necessary, i.e., backpropagation is applicable to any acyclic neural network. Our proof is based on a new formulation of backpropagation called an Acyclic Neural Network (ANN). In ANN, a net is defined as a partially ordered set of processing units where every unit may receive an input value and/or a correction (teaching) value. Therefore, there is no need to differentiate between the input, hidden, and output units.

1. Introduction

The discovery of backpropagation [1][2][3] is a turning point in the history of neural network research. Though it has been suggested by some authors [4][5] that backpropagation is applicable to an arbitrary acyclic neural network, its application is usually restricted to a layered feedforward network, where no lateral connections exist among the processing units of the same layer and no connections bypasses intermediate layers. Mathematical derivations of the rule usually assume the same. Almeida [6] and Pineda [7] showed, in the efforts to extend backpropagation to non-feedforward networks, that backpropagation is applicable to arbitrary feedforward networks. Their proof is indirect, using the theory of linear networks and the concept of transposition. The purpose of this report is to construct a direct algebraic proof. This result would provide neural network designers with more flexibility in their choice of network architecture.

A typical derivation of the backpropagation rule for a layered network is as follows (Figure 1):

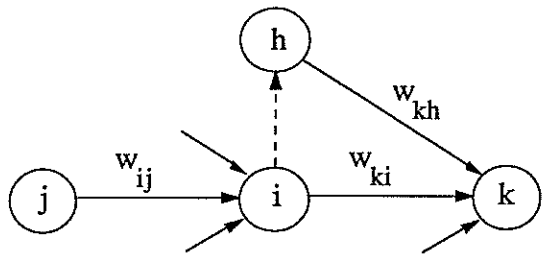

Figure 1: Derivation of Generalized Delta Rule

Let a_i be the activation value of the i-th unit and w_{ij} be the connection weight from the j-th unit to the i-th unit. Let p_i be the net input value to the i-th unit and $\lambda(x)$ be the logistic activation function. Then,

$$a_i \underset{\Delta}{=} \lambda(p_i), \qquad p_i \underset{\Delta}{=} \sum_{j<i} w_{ij} a_j \qquad \text{where} \quad \lambda(x) \underset{\Delta}{=} \frac{1}{1+e^{-x}}. \tag{1.1}$$

In Figure 1, we assume that units j, i and k belong to three different successive layers. The i-th unit and the h-th unit belongs to the same layer but no connection exists between them. Thus, 'j<i' in the summation symbol for p_i should read 'for all j belonging to the previous layer of i'.

Similarly, the activation value of the k-th unit is computed by,

$$a_k \underset{\Delta}{=} \lambda(p_k), \quad p_k \underset{\Delta}{=} \sum_{i<k} w_{ki} a_i . \tag{1.2}$$

Let E be the cost (error) function to be minimized by the gradient descent method. E is a function of the activation values of all of the output units. Then, the learning rule can be derived by computing the following weight modification value:

$$\Delta w_{ij} = -\eta \frac{\partial E}{\partial w_{ij}} = -\eta \frac{\partial E}{\partial p_i} \frac{\partial p_i}{\partial w_{ij}} = -\eta \frac{\partial E}{\partial p_i} a_j = \eta \cdot b_i \cdot a_j \qquad (1.3)$$

where η is the learning rate constant, and

$$b_i \underset{\Delta}{=} -\frac{\partial E}{\partial p_i} = -\frac{\partial E}{\partial a_i} \frac{\partial a_i}{\partial p_i} = -\frac{\partial E}{\partial a_i} \mu(a_i) = q_i \cdot \mu(a_i) \qquad \text{where} \quad \mu(a_i) \underset{\Delta}{=} \frac{\partial a_i}{\partial p_i} = a_i(1 - a_i)$$

$$q_i \underset{\Delta}{=} -\frac{\partial E}{\partial a_i} = \sum_{i<k} -\frac{\partial E}{\partial p_k} \frac{\partial p_k}{\partial a_i} = \sum_{i<k} b_k \frac{\partial p_k}{\partial a_i}. \qquad (1.4)$$

Note that (1.4) is justified because the effect of a change in a_i on E is propagated through all the units in the succeeding layer connected to the i-th unit, and there is no connection, therefore no interaction, among them.

Now, from the definition of p_k in (1.2), with the assumption that no lateral connection exists among the units connected to the k-th unit,

$$\frac{\partial p_k}{\partial a_i} = \sum_{h<k} w_{kh} \frac{\partial a_h}{\partial a_i} = w_{ki} \qquad \because \ \frac{\partial a_h}{\partial a_i} = 0 \quad (h \neq i). \qquad (1.5)$$

Substituting (1.5) into (1.4), we get

$$q_i = \sum_{i<k} w_{ki} b_k. \qquad (1.6)$$

Thus, for a layered network, the rule requires the backward propagation of the error values, b_i's, in a way similar to that in which the activation values, a_i's, are propagated forward.

If, on the other hand, a connection from the i-th unit to the h-th unit exists, as indicated by a dashed arrow in Figure 1, then the computation of q_i is not as simple as in (1.5), because

$$\frac{\partial a_h}{\partial a_i} \neq 0.$$

It is not even clear whether it is sufficient to propagate back the single set of error values. It may be necessary for each unit to compute more than one kind of error value in order to differentiate various backward paths for error propagation.

We will show that that is not the case. The computation of the error value, b_i, remains the same as (1.6) even when there is a connection from the i-th unit to the h-th unit. The only difference is the interpretation of 'i<k' in (1.6). It should read, for an acyclic network, as 'for each unit k which receives the output of unit i directly as an input'. Thus the error value of the k-th unit backpropagates to both the h-th unit and the i-th unit, while the i-th unit receives error values from both the k-th unit and the h-th unit in Figure 1.

In the next section, we will first define the backpropagation network in the most general form. In particular, we eliminate the distinction among the input, hidden, and output units. Through this generalization we underscore the elegant symmetry that exists between the forward propagation of the activation values and the backward propagation of the error values. We will call the new model Acyclic Neural Network (ANN).

In Section 3, we will derive a learning rule for a simple ANN net, in order to demonstrate the basic idea behind our proof.

In Section 4, we will prove the validity of the learning rule given in Section 2.

2. Acyclic Neural Network (ANN)

An ANN net consists of a partially ordered finite set of *processing units* and a finite set of *connections* among them. Each processing unit may receive an *input value* from the environment and may send an *activation value* to the environment. The vector of the input values to the processing units constitute an *input pattern*, and the vector of the activation values constitute an *output pattern*. The purpose of the net is to establish, through learning, a mapping between the input and output patterns defined by a sample training set of input/output pairs. After computing the activation value for the current input value, each processing unit receives a *correction value* from the environment that specifies the difference between the desired activation value (*teaching value*) and the computed activation value. The correction value should be zero if the activation value of the unit is irrelevant. Otherwise, the correction value is a function of the activation value. Each processing unit also computes the *error value* that represents the unit's contribution to the overall difference between the desired output and the actual output patterns.

The activation value of a processing unit is a function of both the activation values of all the predecessor units and the current input value to the unit. The contribution of a predecessor unit is weighted by the *connection weight*. Similarly, the error value of a processing unit is a function of both the error values of all the successor units and the current correction value. The contribution of a successor unit to the error value is also weighted by the connection weight to the successor unit. The net modifies its connection weights after each presentation of a teaching pair so that the mean square sum of the error values of the all processing units is minimized.

Figure 2 illustrates the scheme of an ANN net with n (>0) processing units. The processing units are topologically sorted, and a unit with a smaller index is connected to a unit with a larger index. The input pattern, the output pattern, and the correction values are represented by the vectors, $\{x_i\}$, $\{a_i\}$, and $\{c_i\}$, respectively.

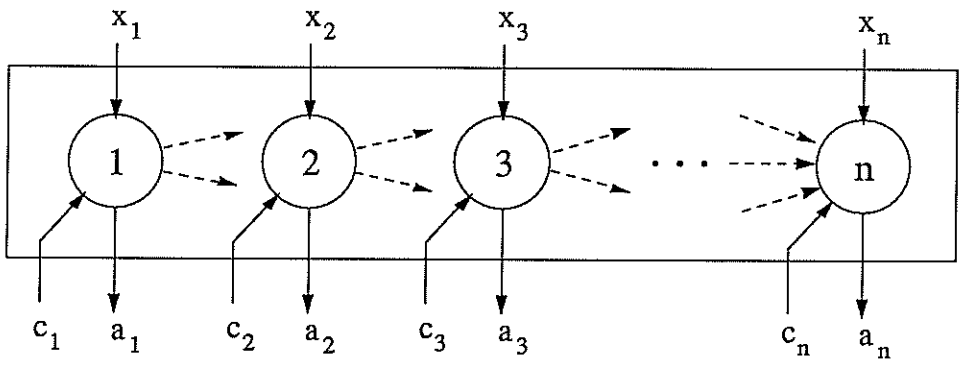

Figure 2: Scheme of an ANN net

We now formally define an ANN net. Let $R = (-\infty, \infty)$, the set of real numbers, and $I = (0,1)$, the set of real numbers between 0 and 1.

Definition: An *ANN net* is a finite acyclic labeled digraph N = (Π, Γ, W) where

$\Pi \equiv \{1, 2, \ldots, n\}$: the set of nodes representing the processing units,

$\Gamma \subseteq \Pi \times \Pi$: the set of arcs representing the connections, such that

for any $i,j \in \Pi$, $(i,j) \in \Gamma \Rightarrow i<j$,

$W : \Gamma \rightarrow R$: the label of the arcs representing the connection weights.

In order to define the behavior of the net N, we use the following values associated with each processing unit, $i \in \Pi$:

$x_i \equiv X(i)$	the *input* value,	where	$X : \Pi \rightarrow R$,
$a_i \equiv A(i)$	the *activation* value,		$A : \Pi \rightarrow I$,
$b_i \equiv B(i)$	the *back error* value,		$B : \Pi \rightarrow R$,
$c_i \equiv C(i)$	the *correction* value,		$C : \Pi \rightarrow (-1, 1)$,
$p_i \equiv P(i)$	the *net input* value,		$P : \Pi \rightarrow R$,
$q_i \equiv Q(i)$	the *net error* value,		$Q : \Pi \rightarrow R$,
$t_i \equiv T(i)$	the *teaching* value,		$T : \Pi \rightarrow I$.

The behavior of N can be divided into two phases, the *forward propagation phase* and the *backward propagation phase*. For a given input value X, the net N computes the activation value A in the forward propagation phase. The difference between the activation value A and the teaching value T is given to the net as the correction value $C = T - A$. In the backward propagation phase the net computes the back error value B from C. Then, using A and B, the net modifies the connection weight W as follows:

Forward: $a_i \overset{\Delta}{=} \lambda(p_i)$, $p_i \overset{\Delta}{=} \sum_{j<i} w_{ij} a_j + x_i$ where $\lambda(x) \overset{\Delta}{=} \frac{1}{1+e^{-x}}$

Backward: $b_i \overset{\Delta}{=} \mu(a_i) \cdot q_i$, $q_i = \sum_{i<k} w_{ki} b_k + c_i$ where $\mu(x) \overset{\Delta}{=} x \cdot (1-x)$

$\Delta w_{ij} \overset{\Delta}{=} \eta \cdot b_i \cdot a_j$ where $\eta \in I$ (the *learning rate*).

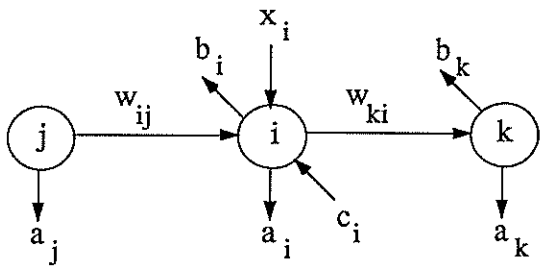

Figure 3: Forward/Backward Propagation

Note that the above definition is a generalization of the multilayered backpropagation network. For example, the backpropagation net of Figure 4 (a) is equivalent to the ANN net of Figure 4 (b).

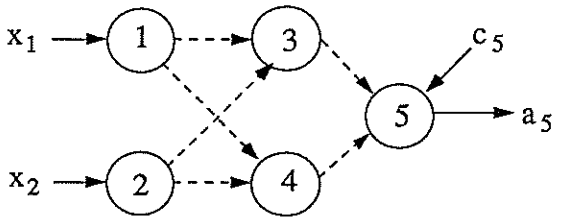

Figure 4 (a): A Backpropagation Net

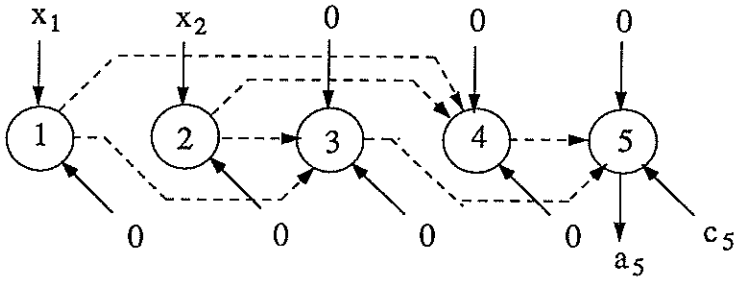

Figure 4 (b): An ANN Representation

3. A Learning Rule for A Simple Net

In order to illustrate the intricacies involved in extending the generalized delta rule to acyclic networks, we will derive a learning rule for the network of Figure 5.

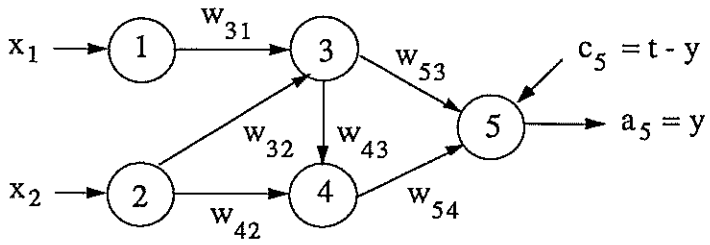

Figure 5: A Simple Net

In the forward propagation phase, the activation values will be computed as follows:

$$\begin{array}{ll} a_1 = \lambda(p_1) & p_1 = x_1 \\ a_2 = \lambda(p_2) & p_2 = x_2 \\ a_3 = \lambda(p_3) & p_3 = w_{31}\, a_1 + w_{32}\, a_2 \\ a_4 = \lambda(p_4) & p_4 = w_{42}\, a_2 + w_{43}\, a_3 \\ a_5 = \lambda(p_5) & p_5 = w_{53}\, a_3 + w_{54}\, a_4. \end{array} \tag{3.1}$$

Let E be the error function defined as:

$$E \underset{\Delta}{=} \frac{1}{2} c_5^2 = \frac{1}{2}(t-y)^2 = \frac{1}{2}(t-a_5)^2. \tag{3.2}$$

Using the definitions given in Section 1,

$$\Delta w_{ij} = \eta \cdot b_i \cdot a_j, \qquad 3 \le i \le 5,\ 1 \le j \le 4, \tag{3.3}$$

where $b_i \underset{\Delta}{=} q_i \cdot \mu(a_i)$,

$$q_i \underset{\Delta}{=} -\frac{\partial E}{\partial a_i} = (t-a_5) \cdot \frac{\partial a_5}{\partial a_i}. \tag{3.4}$$

However, $\dfrac{\partial a_5}{\partial a_5} = 1$ and $\dfrac{\partial a_i}{\partial p_i} = \mu(a_i)$, $1 \le i \le 5$, by the definition of l and m given in Section 2.

Thus, from (3.1),

$$\frac{\partial a_5}{\partial a_i} = \frac{\partial a_5}{\partial p_5}\frac{\partial p_5}{\partial a_i} = \mu(a_5)\left\{ w_{53}\frac{\partial a_3}{\partial a_i} + w_{54}\frac{\partial a_4}{\partial a_i} \right\} \qquad 3 \le i \le 4,$$

i.e., $$\frac{\partial a_5}{\partial a_4} = \mu(a_5)\left\{ w_{53}\frac{\partial a_3}{\partial a_4} + w_{54}\frac{\partial a_4}{\partial a_4}\right\} = \mu(a_5)w_{54} \qquad \because \frac{\partial a_3}{\partial a_4} = 0.$$

Similarly, $$\frac{\partial a_5}{\partial a_3} = \mu(a_5)\left\{ w_{53}\frac{\partial a_3}{\partial a_3} + w_{54}\frac{\partial a_4}{\partial a_3}\right\} = \mu(a_5)\left\{ w_{53} + w_{54}\frac{\partial a_4}{\partial a_3}\right\}, \tag{3.5}$$

and from (3.1), $$\frac{\partial a_4}{\partial a_3} = \frac{\partial a_4}{\partial p_4}\frac{\partial p_4}{\partial a_3} = \mu(a_4)\left\{ w_{42}\frac{\partial a_2}{\partial a_3} + w_{43}\frac{\partial a_3}{\partial a_3}\right\} = \mu(a_4)w_{43} \quad \because \frac{\partial a_2}{\partial a_3} = 0.$$

Therefore,

$$\frac{\partial a_5}{\partial a_4} = \mu(a_5)w_{54} ,$$

$$\frac{\partial a_5}{\partial a_3} = \mu(a_5)w_{53} + \mu(a_5)w_{54}\mu(a_4)w_{43} ,$$

and from(3.4), $q_5 = (t - a_5)\cdot\frac{\partial a_5}{\partial a_5} = (t - a_5) = c_5$ $\qquad b_5 = \mu(a_5)\cdot q_5$

$$q_4 = (t - a_5)\cdot\mu(a_5)w_{54} = c_5\mu(a_5)w_{54} = b_5 w_{54} \qquad b_4 = \mu(a_4)\cdot q_4$$

$$\begin{aligned} q_3 &= (t - a_5)\cdot\{\mu(a_5)w_{53} + \mu(a_5)w_{54}\mu(a_4)w_{43}\} \\ &= c_5\mu(a_5)\{w_{53} + w_{54}\mu(a_4)w_{43}\} \\ &= b_5 w_{53} + b_5 w_{54}\mu(a_4)w_{43} \\ &= b_5 w_{53} + q_4\mu(a_4)w_{43} = b_5 w_{53} + b_4 w_{43} \end{aligned} \qquad b_3 = \mu(a_3)\cdot q_3.$$

In summary, we have the following updating rules for w_{ij}'s, which are exactly the same rules for multilayered backpropagation networks.

$$\begin{aligned} b_5 &= \mu(a_5)\, q_5 & q_5 &= c_5 = t - a_5 \\ b_4 &= \mu(a_4)\, q_4 & q_4 &= w_{54}\, b_5 \\ b_3 &= \mu(a_3)\, q_3 & q_3 &= w_{53}\, b_5 + w_{43}\, b_4 \end{aligned}$$

$$\begin{aligned} \Delta w_{54} &= \eta b_5\, a_4 \\ \Delta w_{53} &= \eta b_5\, a_3 \\ \Delta w_{43} &= \eta b_4\, a_3 \\ \Delta w_{42} &= \eta b_4\, a_2 \\ \Delta w_{32} &= \eta b_3\, a_2 \\ \Delta w_{31} &= \eta b_3\, a_1 \end{aligned}$$

4. The Main Theorem

In this section we prove that the behavior of an ANN net N = (Π, Γ, *W*) defined in Section 2 minimizes the mean square sum of the error values of the all processing units. For simplicity, we assume that the training set is a singleton set. We use the same definitions and notations as in Section 2.

Let $a_i \overset{\Delta}{=} \lambda(p_i)$, where $\lambda(x) \overset{\Delta}{=} \dfrac{1}{1+e^{-x}}$,

$$p_i \overset{\Delta}{=} \sum_{j<i} w_{ij}a_j + x_i\ , \qquad 1\le i\le n, \tag{4.1}$$

and let $E \overset{\Delta}{=} \dfrac{1}{2}\sum_{m=1}^{n} c_m^2 = \dfrac{1}{2}\sum_{m=1}^{n}(t_m - a_m)^2$, where $c_m \overset{\Delta}{=} t_m - a_m$, $1\le m\le n$

be the error function to be minimized by weight modifications. Using the gradient descent method,

$$\Delta w_{ij} = -\eta\frac{\partial E}{\partial w_{ij}} = -\eta\frac{\partial E}{\partial p_i}\frac{\partial p_i}{\partial w_{ij}} = -\eta\frac{\partial E}{\partial p_i}a_j = \eta\cdot b_i\cdot a_j$$

where $b_i \overset{\Delta}{=} -\dfrac{\partial E}{\partial p_i} = -\dfrac{\partial E}{\partial a_i}\dfrac{\partial a_i}{\partial p_i} = -\dfrac{\partial E}{\partial a_i}\mu(a_i) = q_i\cdot\mu(a_i)$, (4.2)

and $q_i \overset{\Delta}{=} -\dfrac{\partial E}{\partial a_i} = (t_i - a_i) + \sum_{i<k}(t_k - a_k)\dfrac{\partial a_k}{\partial a_i} = c_i + \sum_{i<k} c_k\dfrac{\partial a_k}{\partial a_i}$ (4.3)

Note that in derivation of (4.3), we use the fact that if k<i, $\dfrac{\partial a_k}{\partial a_i} = 0$.

Theorem: $q_i = \sum_{i<k} w_{ki}b_k + c_i$, $1\le i\le n$.

Proof: Since the net is acyclic, we can assume that $w_{ij} = 0$ if $i\le j$. Let

$$\boldsymbol{B} \overset{\Delta}{=} \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix} \quad \boldsymbol{C} \overset{\Delta}{=} \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{bmatrix} \quad \boldsymbol{Q} \overset{\Delta}{=} \begin{bmatrix} q_1 \\ q_2 \\ \vdots \\ q_n \end{bmatrix} \quad \boldsymbol{W} \overset{\Delta}{=} \begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1n} \\ w_{21} & w_{22} & \cdots & w_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ w_{n1} & w_{n2} & \cdots & w_{nn} \end{bmatrix} = \begin{bmatrix} 0 & 0 & \cdots & 0 \\ w_{21} & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ w_{n1} & w_{n2} & \cdots & 0 \end{bmatrix}$$

Then, we want to show that $\boldsymbol{Q} = \boldsymbol{W}^{\mathrm{T}}\boldsymbol{B} + \boldsymbol{C}$ i.e., $\begin{bmatrix} q_1 \\ q_2 \\ \vdots \\ q_n \end{bmatrix} = \begin{bmatrix} 0 & w_{21} & \cdots & w_{n1} \\ 0 & 0 & \cdots & w_{n2} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix}\begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix} + \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{bmatrix}$.

Now, let

$$\underline{\underline{L}} \triangleq \begin{bmatrix} \mu(a_1) & 0 & \cdots & 0 \\ 0 & \mu(a_2) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \mu(a_n) \end{bmatrix}$$

$$\underline{\underline{U}} \triangleq \begin{bmatrix} \frac{\partial p_1}{\partial a_1} & \frac{\partial p_1}{\partial a_2} & \cdots & \frac{\partial p_1}{\partial a_n} \\ \frac{\partial p_2}{\partial a_1} & \frac{\partial p_2}{\partial a_2} & \cdots & \frac{\partial p_2}{\partial a_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial p_n}{\partial a_1} & \frac{\partial p_n}{\partial a_2} & \cdots & \frac{\partial p_n}{\partial a_n} \end{bmatrix} \qquad \underline{\underline{V}} \triangleq \begin{bmatrix} \frac{\partial a_1}{\partial a_1} & \frac{\partial a_1}{\partial a_2} & \cdots & \frac{\partial a_1}{\partial a_n} \\ \frac{\partial a_2}{\partial a_1} & \frac{\partial a_2}{\partial a_2} & \cdots & \frac{\partial a_2}{\partial a_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial a_n}{\partial a_1} & \frac{\partial a_n}{\partial a_2} & \cdots & \frac{\partial a_n}{\partial a_n} \end{bmatrix}.$$

Since p_i is independent of a_j for $i \le j$, and similarly, a_i is independent of a_j for $i < j$,

$$\frac{\partial p_i}{\partial a_j} = 0 \text{ if } i \le j, \text{ and } \frac{\partial a_i}{\partial a_j} = 0 \text{ if } i < j.$$

Therefore, $$\underline{\underline{U}} = \begin{bmatrix} 0 & 0 & \cdots & 0 \\ \frac{\partial p_2}{\partial a_1} & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial p_n}{\partial a_1} & \frac{\partial p_n}{\partial a_2} & \cdots & 0 \end{bmatrix} \qquad \underline{\underline{V}} = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ \frac{\partial a_2}{\partial a_1} & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial a_n}{\partial a_1} & \frac{\partial a_n}{\partial a_2} & \cdots & 1 \end{bmatrix}.$$

From (4.1), $\frac{\partial p_i}{\partial a_k} = \sum_{j<i} w_{ij} \frac{\partial a_j}{\partial a_k}$, $\quad k<i, \quad$ i.e.,

$$\begin{bmatrix} 0 & 0 & \cdots & 0 \\ \frac{\partial p_2}{\partial a_1} & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial p_n}{\partial a_1} & \frac{\partial p_n}{\partial a_2} & \cdots & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 & \cdots & 0 \\ w_{21} & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ w_{n1} & w_{n2} & \cdots & 0 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & \cdots & 0 \\ \frac{\partial a_2}{\partial a_1} & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial a_n}{\partial a_1} & \frac{\partial a_n}{\partial a_2} & \cdots & 1 \end{bmatrix}$$

i.e., $$\underline{\underline{U}} = \underline{\underline{W}}\underline{\underline{V}}. \qquad (4.4)$$

Now, since $\frac{\partial a_i}{\partial a_j} = \frac{\partial a_i}{\partial p_i}\frac{\partial p_i}{\partial a_j} = \mu(a_i)\frac{\partial p_i}{\partial a_j}$, for j<i, i.e.,

$$\begin{bmatrix} 0 & 0 & \cdots & 0 \\ \frac{\partial a_2}{\partial a_1} & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial a_n}{\partial a_1} & \frac{\partial a_n}{\partial a_2} & \cdots & 0 \end{bmatrix} = \begin{bmatrix} \mu(a_1) & 0 & \cdots & 0 \\ 0 & \mu(a_2) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \mu(a_n) \end{bmatrix} \times \begin{bmatrix} 0 & 0 & \cdots & 0 \\ \frac{\partial p_2}{\partial a_1} & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial p_n}{\partial a_1} & \frac{\partial p_n}{\partial a_2} & \cdots & 0 \end{bmatrix}$$

we get $(\boldsymbol{V} - \boldsymbol{I}) = \boldsymbol{LU}$ where I is the unit matrix. (4.5)

From (4.2), $\boldsymbol{B} = \boldsymbol{LQ}$ (4.6)

and from (4.3), $\boldsymbol{Q} = \boldsymbol{V}^{\mathrm{T}}\boldsymbol{C}$, i.e., (4.7)

$$\begin{bmatrix} q_1 \\ q_2 \\ \vdots \\ q_n \end{bmatrix} = \begin{bmatrix} 1 & \frac{\partial a_2}{\partial a_1} & \cdots & \frac{\partial a_n}{\partial a_1} \\ 0 & 1 & \cdots & \frac{\partial a_n}{\partial a2} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix} \times \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{bmatrix}$$

We want to show that from (4.1) - (4.7) we can derive $\boldsymbol{Q} = \boldsymbol{W}^{\mathrm{T}}\boldsymbol{B} + \boldsymbol{C}$.

From (4.4) and (4.5),

$$\boldsymbol{V} = (\boldsymbol{LW})\boldsymbol{V} + \boldsymbol{I} = \boldsymbol{RV} + \boldsymbol{I} \tag{4.8}$$

where $\boldsymbol{R} \underset{\Delta}{=} \boldsymbol{LW} = \begin{bmatrix} 0 & 0 & \cdots & 0 \\ \mu(a_2)w_{21} & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \mu(a_n)w_{n1} & \mu(a_n)w_{n2} & \cdots & 0 \end{bmatrix}$ (4.9)

Then, since $\boldsymbol{R}^m = \boldsymbol{0}$ when n≤m, from (4.8),

$$\boldsymbol{V} = \boldsymbol{R}^* = \sum_{i=0}^{n-1} \boldsymbol{R}^i$$

$$\because \quad RV + I = R(I + R + \cdots + R^{n-1}) + I$$
$$= I + R + \cdots + R^{n-1} + R^{n}$$
$$= V$$

Since $V^{T} = (R^{*})^{T} = (R^{T})^{*}$,

from (4.7), $Q = V^{T}C = (R^{T})^{*}C = (I^{T} + R^{T} + \cdots + (R^{T})^{n-1})C$

$$\therefore \quad Q = R^{T}Q + C \tag{4.10}$$

Finally, substituting (4.9) into (4.10),

$$Q = (LW)^{T}Q + C$$
$$= W^{T}L^{T}Q + C$$
$$= W^{T}LQ + C \qquad \because L = L^{T}$$
$$= W^{T}B + C \qquad \text{from (4.6).}$$

Q.E.D.

5. References

[1] Le Cun, Y. "A learning scheme for asymmetric threshold network," in *Cognitiva 85'*, CESTA-AFCET (ed.), 1985, pp. 559-604.

[2] Werbos, P. *Beyond regression: new tools for prediction and analysis in behavioral sciences*, Ph.D. Thesis, Harvard University, 1984.

[3] Rumelhart, D. E. & McClelland, J. L. *Parallel Distributed Processing*, Vol. 1 and 2, MIT Press, 1986.

[4] Soulie, F. F., Gallinari, P., Le Cun, Y., and Thiria, S. "Automata networks and artificial intelligence," in *Automata networks in computer science*, F. F. Soulie, Y. Robert, and M. Tchuente (ed.), Princeton University Press, 1987.

[5] Arbib, M. A. *Brains, Machines, and Mathematics*, Second Edition, Springer-Verlag, 1987.

[6] Almeida, L. B. "Backpropagation in non-feedforward networks," in *Nueral Computing Architectures*, I. Aleksander (ed.), The MIT Press, 1989.

[7] Pineda, J. "Generalization of backpropagation to recurrent neural networks," *Proc. of the IEEE Conference on Neural Information Processing Systems - Natural and Synthetic*, Boulder, CO, November 1987.